# Requirements Traceability Link Recovery via Retrieval-Augmented Generation

Tobias Hey[0000−0003−0381−1020], Dominik Fuchß[0000−0001−6410−6769], Jan Keim[0000−0002−8899−7081], and Anne Koziolek[0000−0002−1593−3394]

KASTEL – Institute of Information Security and Dependability
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{hey, dominik.fuchss, jan.keim, koziolek}@kit.edu

**Abstract.** [**Context and Motivation**] In software development, various interrelated artifacts are created. Access to information on the relation between these artifacts eases understanding of the system and enables tasks such as change impact and software reusability analyses. Manual trace link creation is labor-intensive and costly, and thus is often missing in projects. Automation could enhance the development and maintenance efficiency. [**Question/Problem**] Current methods for automatically recovering traceability links between different types of requirements do not achieve the necessary performance to be applied in practice, or require pre-existing links for machine learning. [**Principal Ideas and Results**] We propose to address this limitation by leveraging large language models (LLMs) with retrieval-augmented generation (RAG) for inter-requirements traceability link recovery. In an empirical evaluation on six benchmark datasets, we show that chain-of-thought prompting can be beneficial, open-source models perform comparably to proprietary ones, and that the approach can outperform state-of-the-art and baseline approaches. [**Contribution**] This work presents an approach for inter-requirements traceability link recovery using RAG and provides the first empirical evidence of its performance.

**Keywords:** Traceability Link Recovery · Requirements Traceability · Requirements Engineering · LLM · Retrieval-Augmented Generation

## 1 Introduction

During software system development and maintenance, different artifacts are created, modified, and maintained. These artifacts are interrelated, typically representing refinements, implementations, or views of another artifact. Access to the information that artifacts are related to each other eases understanding the system and aids in tasks such as assessing the impact of changes [5,40]. However, manually creating or maintaining this traceability information is labor-intensive and costly. Therefore, in most software projects this information is not readily available. Automating the generation of traceability information could improve the efficiency of software development, maintenance, and management in many

projects that currently do not have access to this kind of information [5, 13]. Additionally, regulatory bodies mandate traceability in safety-critical systems such as aerospace or automotive [10, 31, 34].

Existing approaches for automatically recovering trace links between different types of requirements still do not achieve the necessary performance to fully automate the task [1, 16, 39] or need initial links in a project to fit a machine learning (ML)-based approach to the project specifics [14, 28, 29, 41]. However, as an unsupervised traceability link recovery (TLR) without the need for initial links would be able to provide traceability information to a larger group of projects, we target this task in our work.

Recently, large language models (LLMs) have shown promising performance on related software engineering tasks, such as bug localization [22, 26], issue-commit linking [20, 23], and code search [3]. As they show impressive performance, especially in understanding and analyzing natural language documents, and the fact that inter-requirements traceability mainly deals with those types of documents, we want to investigate their potential for the task.

However, pretrained LLMs may not have access to the project specifics and its contexts out-of-the-box. Simply presenting all potential trace link candidates to the LLM might not be possible, as their input length is limited. Fortunately, the processing can be augmented by retrieval techniques, called retrieval-augmented generation (RAG) [21]. In RAG, relevant documents to a query are first retrieved based on information retrieval (IR). Then, the LLM is tasked to generate the answer by taking into account the retrieved documents.

Therefore, we propose to use RAG for traceability link recovery between requirements. We investigate the potential of RAG for inter-requirements TLR and put the results into perspective by comparing them to state-of-the-art and baseline approaches. This leads us to our first research question:

**RQ1**: How does a RAG-based TLR approach perform compared to baseline and state-of-the-art approaches?

As recent research indicates that chain-of-thought (CoT) prompting can be more effective than other zero-shot prompts [33], we further explore how the performance is affected by this prompting technique:

**RQ2**: Does chain-of-thought prompting improve the performance compared to a simple classification prompt?

Access to an open-source and self-hostable option of the LLMs may make the approach applicable to a wider range of projects. Therefore, we additionally investigate the following research question:

**RQ3**: Is the performance of current open-source LLMs comparable to the performance of proprietary ones?

Our contributions are insights into the performance of a RAG-based approach for TLR between requirements, an empirical study on the effect of different LLMs and prompt techniques on the performance of this RAG-based approach, and a comparison to existing automated inter-requirements TLR approaches.

## 2 Related Work

Automatically recovering trace links between requirements has been in the focus of research for the past two decades. Early approaches utilize classical IR techniques such as vector space model (VSM) [15], latent semantic indexing (LSI) [16], and the probabilistic network model [43]. In addition to simply using the requirements as inputs, approaches explored the use of the project glossary [43], thesauri [15,27] and ontologies [2]. Other approaches combine multiple IR techniques [25] or techniques for directly linking certain artifact types with techniques using intermediate artifacts [32].

More recently, approaches utilize word embeddings [4, 17, 42] and machine learning [14, 28, 29, 41] to tackle the task. However, besides S2Trace by Chen et al. [4], these approaches require initial links between the artifacts to train the models, which is a different task than tackled in this paper.

Schlutter and Vogelsang [37, 38] use spreading activation on a combination of dependency and semantic role label graphs to identify trace links between different types of requirements.

Current approaches use LLMs to recover trace links. Lin et al. [24] explore the performance of domain-adapted BERT-LLMs. They compare a generic software engineering LLM, a project domain-adapted version, and a trace task-related adaptation. In their experiments, the trace task-related training on issue-commit links in GitHub performed best on the tasks of trace link completion and expansion. Although their approach targeted these tasks they also measured the zero-shot performance of their Task-CLS and Task-RANK models on generating trace links without existing initial links, which is a similar task to ours.

Rodriguez et al. [33] presented a preliminary study on applying prompt engineering to TLR. On a subset of the CM1 and Dronology [6] dataset, they explored the performance of different prompts and prompt types showing that small prompt modifications can result in differences in TLR results. Chain-of-thought prompting improved both the precision and recall of the recovery.

Preda et al. [30] use GPT-3.5 and GPT-4 models to detect whether a given set of low-level requirements fulfills the aspects described in the corresponding high-level requirements. They could show that zero-shot prompting by asking the LLM to explain their answer performed best on the task.

## 3 Approach

This section presents the RAG-based approach and discusses its different components. Retrieval-augmented generation (RAG) typically consists of two steps: first, identifying relevant documents via IR techniques and second, prompting the LLM to perform the actual task in the context of the retrieved documents. In our case, relevant documents are requirements from the set of target artifacts that might be candidates for having a trace link to a certain source requirement. The approach takes two sets of requirements files as input: the set of source requirements (e.g., high-level requirements) and the set of target requirements

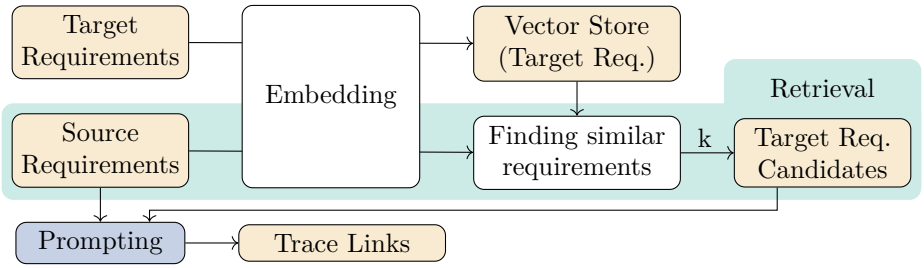


**Fig. 1.** Overview of the retrieval and prompting steps with data in orange, prompting in blue, and the other processing in white.

(e.g., low-level requirements). For the semantics of such a trace link, we assume that they are in a refinement or predecessor-successor relationship. The output of the approach is a list of trace links between the requirements in the two sets.

### 3.1 Retrieval Step

To retrieve relevant documents, we use document embeddings. We transform both the source and target requirements using the embedding model of an LLM into a vector representation. Since the requirements can typically be regarded as shorter texts, we directly apply the embedding models to a whole requirement. The embeddings of the requirements from the target set are then stored in a vector store. For each source requirement, we then retrieve potential candidates for the prompting, as shown in Figure 1.

Thus, the step is responsible for finding the Top-$k$ most similar target requirements in the vector store for a given source requirement. We use the cosine similarity of the requirements' embedding vectors to calculate similarity. These $k$ target requirements are used as trace link candidates for the prompting step.

### 3.2 Prompting Step

In the prompting step, for each pair consisting of a source requirement and a target requirement candidate from the retrieval step, a prompt for the LLM is created. Depending on the response of the LLM, the approach decides if it should create a trace link between the source requirement and the candidate target requirement.

In principle, the approach supports different prompting techniques. However, in this paper, we do not focus on prompt engineering, i.e., we do not extensively modify prompts and compare their performance. Instead, we build on the findings of Rodriguez et al. [33], from which we derive two prompt types: a KISS prompt and a chain-of-thought (CoT) prompt. Both prompts also performed well for other TLR tasks [11]. The KISS prompt serves as a straightforward classification prompt, while we designed the CoT prompt to ask the LLM to reason about the artifacts.

*KISS Prompt* The KISS prompt is a simple zero-shot prompt (cf. Prompt 1). The prompt first states that the software development domain is the context. Afterward, the prompt provides the types of artifacts and their contents. The actual question posed to the LLM is then a yes-or-no question if the source requirement is related to the target requirement candidate. In the LLM's response, the approach looks for the term *yes*. If found, the approach creates a trace link between the source and target requirement.

**Prompt 1: KISS**
Question: Here are two parts of software development artifacts.
{source_type}: '''{source_content}'''
{target_type}: '''{target_content}'''
Are they related?
Answer with 'yes' or 'no'.

*Chain-of-thought Prompt* This prompt uses chain-of-thought (CoT) prompting to decide the traceability (cf. Prompt 2). It first sets the context for the LLM by defining that it will get two artifacts from the same software system. Afterward, the prompt asks if there should be a traceability link between the two artifacts. This question is also a simple yes-or-no question as in Prompt 1, but additionally, the LLM is asked to provide reasoning. Lastly, the prompt defines the output format and provides the source and target requirements. Again, the approach analyzes the output of the LLM and creates a trace link between the source requirement and the target requirement if the LLM replied with *yes*.

**Prompt 2: CoT**
Below are two artifacts from the same software system. Is there a traceability link between (1) and (2)? Give your reasoning and then answer with 'yes' or 'no' enclosed in <trace> </trace>.
(1) {source_type}: '''{source_content}'''
(2) {target_type}: '''{target_content}'''

## 4 Evaluation

In this section, we present the empirical evaluation of the approach. We compare the performance of the different prompts and models on six benchmark datasets. Moreover, we relate to the performance of state-of-the-art and baseline approaches.

### 4.1 Experimental Setup

First, we describe the experimental setup we used for our evaluation. In all of our experiments, we set the temperature of the LLMs to zero and set a fixed

**Table 1.** Overview of the datasets. Datasets comprise either high-level requirements (HLR), low-level requirements (LLR), requirements (R) or regulatory codes (RC). Percentage of linked source and target artifacts in the gold standard is given in brackets.

| | Artifact Type | | Number of Artifacts | | | | |
|---|---|---|---|---|---|---|---|
| Dataset | Source | Target | Source | | Target | | TLs |
| CCHIT-2-WorldVista | R | RC | 116 | (62%) | 1064 | (39%) | 587 |
| CM1-NASA | HLR | LLR | 22 | (86%) | 53 | (57%) | 45 |
| Dronology | HLR | LLR | 99 | (96%) | 211 | (99%) | 220 |
| GANNT | HLR | LLR | 17 | (100%) | 69 | (99%) | 68 |
| MODIS | HLR | LLR | 19 | (63%) | 49 | (63%) | 41 |
| WARC | HLR | LLR | 63 | (95%) | 89 | (89%) | 136 |

random seed (133742243) for the LLMs. This functionality makes it possible to achieve deterministic results from the LLMs. We set the number of retrieved trace link candidates to four. We chose $k = 4$, because it represents the average ratio of source artifacts to target artifacts in the datasets. Moreover, in our experiments the average number of target requirements that relate to a certain source requirement always stayed under four.

**Datasets** To compare our results to other approaches, we use datasets that are commonly used in the TLR community. Table 1 gives an overview of the datasets. We took the CCHIT-2-WorldVista, CM1-NASA, GANNT, MODIS, and WARC datasets from the repository of the *Center of Excellence for Software & Systems Traceability* (CoEST) [7]. Additionally, we make use of the Dronology dataset [6]. CM1, Dronology, GANNT, MODIS, and WARC comprise high-level requirements (HLR) and their respective low-level requirements (LLR), whereas CCHIT links requirements to regulatory codes. CCHIT and Dronology are the largest datasets with more than 300 requirements.

However, the gold standard trace links of CCHIT only cover 39% of the 1064 LLR. The gold standards for the CM1 and the MODIS dataset also cover only a low number of artifacts. This can mean two things: either the set of source/target requirements is incomplete, resulting in source requirements not having a link to a target artifact or target artifacts not having a corresponding source artifact in the dataset, or the gold standard is incomplete. Especially for IR-based approaches, this can influence the performance, as they seek to map a source artifact to its most similar target artifacts under the assumption that ideally, each artifact should have a corresponding artifact in the TLR task. On these datasets, previous works achieved a particularly low performance [4,15,42].

**Metrics** As our approach regards the task of recovering trace links as a classification task, we make use of classification metrics that are commonly applied to evaluate the performance of TLR approaches [16, 39]: precision, recall, and the

$F_\beta$-score. We choose $\beta = 1$, as for the goal to fully automate TLR, both precision and recall have to be high [16]. If one weakens the goal to semi-automatic TLR, recall should be valued higher to prevent too many missed links. Therefore, we additionally provide $F_2$-score.

**Large Language Models** In this evaluation, we use different LLMs based on recent state-of-the-art models. We use *text-embedding-3-large* by OpenAI as the embedding model. We opted for that model, as the vendor states it is their newest and best-performing embedding model. We did not include further embeddings in this work to reduce the degrees of freedom.

Regarding the LLMs for prompting, we decided to use models by OpenAI and locally deployed open-source models. For OpenAI, we have chosen recent state-of-the-art models. Namely, we use *GPT-4o mini* and *GPT-4o*. As open-source models, we used *Codellama 13b* and *Meta AI Llama3.1 8b*. We choose the models based on their performance and required resources. Compared to the other models, *Codellama 13b* has been trained to generate and discuss code. The other models are general-purpose models. We used a machine with a Tesla V100S GPU with 32GB VRAM, 112 CPUs (2.7GHz), and 256GB RAM to execute the experiments with the open-source models. For the GPT-based models, we used OpenAI's API.

**Baselines** To evaluate the performance of our approach, we compare it to state-of-the-art and baseline approaches. The first baseline is the IR-only approach. This approach does not use the classifying LLM, but only the retrieval step of our approach. We directly create trace links between a source requirement and all retrieved target requirements candidates. By doing so, we give an upper boundary of the achievable recall of our RAG-based approach, as the prompting can only link a source requirement to target requirements within the top $k$ target requirements retrieved by the retrieval step. This allows us to analyze the effect the classifier step has and whether the classifier is beneficial.

Additionally, we provide the results of two baseline approaches using VSM and LSI as provided by Gao et al. [12] in their replication package. As the approaches only output ranked lists per source artifact and do not define a fixed threshold to determine final trace links, we calculate the optimized $F_1$-score per project by varying the cutoff threshold between $[0, 1]$ in 0.01 steps. Thus, the comparison reflects the upper boundary of the baselines' performance.

For the comparison to existing approaches, we searched for related work that either reported results on the same datasets or provided replication packages to apply their approaches to our datasets. Unfortunately, none of the unsupervised inter-requirements TLR approaches mentioned in Section 2 provided a still accessible replication package, leaving us with a comparison to values reported in the respective publications. Most older approaches only show precision-recall graphs from which it would be imprecise to derive exact numbers, or they require manually provided information such as thesauri and glossaries. Therefore, we only compare our approach with the more recent approaches WQI by Zhao

et al. [42], S2Trace by Chen et al. [4], and Task-CLS/Task-RANK by Lin et al. [24], as they reported either $F_1$- or $F_2$-score on some of the datasets.

**Statistical Significance Tests** For performing statistical significance tests, we follow the guidelines of Dell'Anna et al. [9] for the evaluation of classifiers in software engineering research. As we compare more than two classifiers in this work, we use Friedman's omnibus test with Nemenyi's post-hoc. For measuring the practical significance, we calculate Kendall's W for the effect size of the Friedman test. To interpret the results, we follow Cohen's guidelines [8]: Small effect $[0.1, 0.3)$, medium effect $[0.3, 0.5)$, and large effect $[0.5, 1.0]$.

### 4.2 Results

Before analyzing the performance of our approach in comparison to baselines and state-of-the-art (RQ1), we first investigate the influence of the prompting technique (RQ2) and the chosen LLM (RQ3) on the performance of our approach. Table 2 displays the results in precision (P.), recall (R.), $F_1$- and $F_2$-score on the five datasets and on average. We report the values for the IR-only baseline and the four LLMs with both the KISS and the CoT prompt.

**RQ2: Does chain-of-thought prompting improve the performance in comparison to a simple classification prompt?** Regarding RQ2, the results show that in average $F_1$-score, both GPT models and the Llama model perform better with CoT prompting. Only Codellama achieves a higher $F_1$-score with the KISS prompt. The CoT version of Codellama is also the only variant that performs worse in $F_1$-score than the IR-only baseline. GPT-4o with chain-of-thought prompting achieves the best average performance in precision, $F_1$- and $F_2$-score. With an average $F_1$-score of 45.1%, the improvement is statistically significant (at the $\alpha = 0.05$ level) compared to the IR-only baseline.

However, neither in $F_1$-score nor $F_2$-score the differences between the CoT prompt versions and the KISS prompt versions are statistically significant (at the $\alpha = 0.05$ level). The effect size is small in both cases.

This leads us to the conclusion:

> **Conclusion RQ2:** Although there are no statistically significant differences between the prompt types, it is noticeable that if precision is as important as recall ($F_1$-score), chain-of-thought prompting can improve the performance on inter-requirements traceability link recovery.

Interestingly, both GPT-4o and GPT-4o mini with KISS prompt achieve the same average recall as the IR-only baseline that by design sets the upper boundary of our approach's achievable recall. They also improve precision, which means they can identify some false positives in the target candidate list without discarding correct links. However, the improvements in precision are relatively

**Table 2.** Detailed results of our RAG-based approach with different models and prompt types. Best results per row are highlighted in bold type.

| Dataset | Metric | IR-only | KISS | | | | CoT | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | GPT-4o | GPT-4o mini | Code-llama | Llama 3.1 | GPT-4o | GPT-4o mini | Code-llama | Llama 3.1 |
| CCHIT | P. | .198 | .234 | .212 | .247 | .205 | **.367** | .264 | .206 | .276 |
| | R. | **.157** | **.157** | **.157** | .155 | **.157** | .138 | .145 | .152 | .129 |
| | $F_1$ | .175 | .188 | .180 | .190 | .178 | **.200** | .187 | .175 | .176 |
| | $F_2$ | .164 | **.168** | .165 | .167 | .164 | .158 | .159 | .160 | .145 |
| CM1 | P. | .330 | .341 | .330 | .341 | .330 | **.458** | .358 | .438 | .414 |
| | R. | **.644** | **.644** | **.644** | **.644** | **.644** | .600 | **.644** | .467 | **.644** |
| | $F_1$ | .436 | .446 | .436 | .446 | .436 | **.519** | .460 | .452 | .504 |
| | $F_2$ | .541 | .547 | .541 | .547 | .541 | .565 | .556 | .461 | **.580** |
| Dronology | P. | .386 | .394 | .386 | .417 | .383 | **.512** | .421 | .388 | .405 |
| | R. | **.695** | **.695** | **.695** | .614 | .686 | .655 | .668 | .632 | .568 |
| | $F_1$ | .497 | .503 | .497 | .496 | .492 | **.575** | .517 | .481 | .473 |
| | $F_2$ | .600 | .603 | .600 | .561 | .593 | **.620** | .598 | .561 | .526 |
| GANNT | P. | .544 | .544 | .544 | .544 | .561 | **.607** | .569 | .565 | .571 |
| | R. | **.544** | **.544** | **.544** | **.544** | **.544** | **.544** | **.544** | .515 | .529 |
| | $F_1$ | .544 | .544 | .544 | .544 | .552 | **.574** | .556 | .538 | .550 |
| | $F_2$ | .544 | .544 | .544 | .544 | .547 | **.556** | .549 | .524 | .537 |
| MODIS | P. | .145 | .200 | .193 | .268 | .220 | **.500** | .263 | .153 | .233 |
| | R. | **.268** | **.268** | **.268** | **.268** | **.268** | .171 | .244 | .220 | .244 |
| | $F_1$ | .188 | .229 | .224 | **.268** | .242 | .255 | .253 | .180 | .238 |
| | $F_2$ | .229 | .251 | .249 | **.268** | .257 | .197 | .248 | .202 | .242 |
| WARC | P. | .373 | .387 | .373 | .428 | .374 | **.537** | .408 | .405 | .433 |
| | R. | **.691** | **.691** | **.691** | .566 | .676 | .640 | .654 | .625 | .640 |
| | $F_1$ | .485 | .496 | .485 | .487 | .482 | **.584** | .503 | .491 | .516 |
| | $F_2$ | .590 | .597 | .590 | .532 | .582 | **.616** | .584 | .564 | .584 |
| Avg. | P. | .329 | .350 | .340 | .374 | .345 | **.497** | .381 | .359 | .389 |
| | R. | **.500** | **.500** | **.500** | .465 | .496 | .458 | .483 | .435 | .459 |
| | $F_1$ | .387 | .401 | .394 | .405 | .397 | **.451** | .413 | .386 | .410 |
| | $F_2$ | .445 | **.452** | .448 | .437 | .447 | **.452** | .449 | .412 | .436 |

small in comparison to the GPT-4o CoT variant. The KISS prompt variants usually have the same recall as the IR-only baseline on most projects, but they are less precise than the CoT prompt variants.

**RQ3: Is the performance of current open-source LLMs comparable to the performance of proprietary ones?** If we look at the performance of the open-source models, Codellama and Llama3.1, the results differ in comparison to the proprietary model depending on the type of prompt used. Using the KISS prompt, the open-source models achieve a lower average recall and a higher or comparable precision than the GPT-based models. If we value precision as important as recall ($F_1$-score) the Codellama model performs best with the KISS prompt (40.5%). However, regarding $F_2$-score, both GPT models perform better. With the CoT prompt, however, both open-source models are outperformed by the proprietary models in both average $F_1$-score and $F_2$-score. Overall, the GPT models achieve the best average $F_1$-score and $F_2$-score. However, on the MODIS dataset, the Codellama model performs best.

To answer the question, if the performance between open-source and proprietary LLMs is comparable, we compare the results of the different models to each other with Friedman's omnibus test with Nemenyi's post-hoc. In $F_2$-scores no statistically significant differences can be observed. In $F_1$-score, GPT-4o performs significantly better than both open-source models with medium effect size. GPT-4o mini, however, shows no significant difference to any of the tested models. Therefore, we come to the following conclusion:

> **Conclusion RQ3:** In general open-source and proprietary LLMs perform comparably in inter-requirements traceability link recovery. However, if we rate precision as important as recall ($F_1$-score), at least GPT-4o is able to statistically outperform the open-source models.

**RQ1: How does a RAG-based TLR approach perform compared to baseline and state-of-the-art approaches?** Table 3 shows the results in $F_1$-score and $F_2$-score of our best-performing variants compared to the VSM, LSI, and IR-only baselines. We additionally include the values of WQI, S2Trace, Task-CLS, and Task-RANK as reported in the publications. As discussed in Section 4.1, we could not replicate their or any other approach and, thus, cannot give the full overview of their performance on these datasets. However, most older publications also use VSM and LSI and, thus, perform similarly to our baselines.

If we look at $F_1$-score, the GPT-4o model with CoT prompting outperforms all baselines and the state-of-the-art approaches on all but the MODIS dataset. Compared to the LSI and the IR-only baseline, the improvement is even statistically significant (at the $\alpha = 0.05$ level). Regarding practical significance the effect size is medium (Kendall's $W = 0.415$). On MODIS, however, the VSM baseline achieves an $F_1$-score of 33.3%, 7.8 percentage points higher than GPT-4o with CoT with 25.5%. As on average GPT-4o with CoT achieves a 11.2 percentage

**Table 3.** Comparison to baselines and existing approaches. Our approaches are underlined and the best results per column are highlighted in bold type.

| | | CCHIT | | CM1 | | Dronol. | | GANNT | | MODIS | | WARC | | Avg. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Approach | $F_1$ | $F_2$ | $F_1$ | $F_2$ | $F_1$ | $F_2$ | $F_1$ | $F_2$ | $F_1$ | $F_2$ | $F_1$ | $F_2$ | $F_1$ | $F_2$ |
| | WQI | — | — | .349 | .337 | — | — | .351 | .453 | — | — | — | — | — | — |
| | S2Trace | — | — | .378 | .435 | — | — | .381 | .463 | — | — | — | — | — | — |
| | Task-CLS | — | — | — | **.605** | — | .493 | — | — | — | — | — | — | — | — |
| | Task-RANK | — | — | — | .520 | — | .510 | — | — | — | — | — | — | — | — |
| Baseline | VSM | .190 | **.259** | .429 | .451 | .519 | .552 | .344 | .374 | **.333** | .273 | .221 | .303 | .339 | .369 |
| Baseline | LSI | .186 | .203 | .374 | .413 | .487 | .503 | .326 | .387 | .292 | **.307** | .165 | .207 | .305 | .337 |
| Baseline | IR-only | .175 | .164 | .436 | .541 | .497 | .600 | .544 | .544 | .188 | .229 | .485 | .590 | .387 | .445 |
| KISS | GPT-4o | .188 | .168 | .446 | .547 | .503 | .603 | .544 | .544 | .229 | .251 | .496 | .597 | .401 | **.452** |
| KISS | Codellama | .190 | .167 | .446 | .547 | .496 | .561 | .544 | .544 | .268 | .268 | .487 | .532 | .405 | .437 |
| CoT | GPT-4o | **.200** | .158 | **.519** | .565 | **.575** | **.620** | **.574** | **.556** | .255 | .197 | **.584** | **.616** | **.451** | **.452** |
| CoT | Llama3.1 | .176 | .145 | .504 | .580 | .473 | .526 | .550 | .537 | .238 | .242 | .516 | .584 | .410 | .436 |

points higher $F_1$-score, we still regard our approach as superior to plain VSM. Compared to the non-baseline approaches WQI and S2Trace, which use word embedding-based information retrieval, all our variants and even the IR-only baseline perform better in $F_1$-score. On CM1, GPT-4o with CoT outperforms the better-performing S2Trace by 14.1 percentage points and on GANNT even by 17.5 percentage points. As Lin et al. [24] did not report $F_1$-score, we can only compare to their approaches in $F_2$-score.

In $F_2$-score, all our approaches outperform Task-CLS and Task-RANK on Dronology. On CM1, however, Task-CLS performs best in $F_2$-score with 60.5%. Our best approach achieves 58% and all our variants outperform the 52% of Task-RANK. On average $F_2$-score on CM1 and Dronology GPT-4o with CoT still performs better than all comparison approaches. Compared to WQI and S2Trace on CM1 and GANNT, all our approaches achieve higher $F_2$-scores. Regarding the baselines, GPT-4o with CoT performs, with an average $F_2$-score of 45.2%, 8.3 percentage points better than the VSM (36.9%) and 11.5 percentage points better than the LSI baseline (33.7%). However, due to the lower performance on CCHIT and MODIS, the improvement is not statistically significant in a Wilcoxon signed-rank test.

As our best-performing approach GPT-4o with CoT outperforms all other approaches in average $F_1$- and $F_2$-score, we come to the following conclusion:

**Conclusion RQ1:** In inter-requirements TLR, a RAG-based approach performs better than state-of-the-art and baseline approaches.

### 4.3 Threats to Validity

Based on the guidelines of Runeson and Höst [35] and the threats of using LLMs in software engineering discussed by Sallou et al. [36], we address potential threats of our research and experiments.

*Construct validity* In this work, we explicitly did not focus on prompt engineering, as we aim to provide first exploratory results on RAG for TLR. Thus, a potential threat to the validity of our results is the selection of the prompts used, e.g., the KISS prompt just uses *related* without defining its meaning. However, we derive the prompts from the work of Rodriguez et al. [33].

*Internal validity* Our results and, thus, our conclusions might be influenced by other factors. To mitigate this threat, we follow established practices. We used established benchmark datasets and stated the origins of the projects. This reduces the risk of selection bias.

*External validity* We evaluate our approach on a limited number of projects. The results can vary for other projects. For example, the choice of retrieved trace links candidates $k$ could influence the performance on projects with other characteristics. To reduce this threat, we use benchmark datasets commonly used in TLR research. The datasets cover different domains and project sizes. Nevertheless, most datasets are quite old and might not reflect the challenges for traceability in modern projects. Second, we evaluate the approach with only one embedding model and four LLMs, thus, we cannot state if the results hold for other models. To mitigate this threat, we make use of state-of-the-art models. The third threat is using *closed source models* for evaluation. Since we do not know the training data of these models, we cannot ensure that there is no *data leakage*. The fourth threat is the non-determinism of the LLMs. To mitigate this threat, we set a random seed for the LLMs and set their temperature to 0. Thus, our results are dependent on the chosen seed 133742243.

## 5 Conclusion

In conclusion, this work investigated the capabilities of retrieval-augmented generation (RAG) for automated inter-requirements traceability link recovery (TLR). We presented a two-step approach that first uses embedding-based information retrieval to identify potential trace link candidates (target requirements) for a given source requirement. Then the approach uses those candidates to prompt a generative large language model for identifying the links. We implemented two kinds of prompts: a simple zero-shot prompt (KISS), and a zero-shot prompt using chain-of-thought (CoT).

In an empirical evaluation with six benchmark datasets from the TLR community with four different LLMs, we could show that the approach can outperform state-of-the-art and baseline approaches. On average, our best-performing configuration using GPT-4o and CoT prompting achieved an $F_1$-score of 45.1%

and an $F_2$-score of 45.2%. Furthermore, we observed that CoT can be beneficial in comparison to the KISS prompt, and that current open-source models perform comparably to proprietary ones. However, 45% $F_1$-/$F_2$-score is still insufficient to be used in a fully automated setting in practice.

Consequently, as a next step, we plan to investigate further prompting techniques, such as few-shot prompting and automated prompt engineering. We intend to improve the RAG-based approach by including context information beyond the information in the artifacts. As we have shown in previous work [19] that identifying relevant parts of the requirements for TLR can be beneficial, we will explore the use of requirements classification for this approach as well.

**Data Availability Statement** The code, datasets, statistical tests, and results are publicly available in our replication package [18].

**Acknowledgments.** This work was supported by funding from the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF) and supported by the German Research Foundation (DFG) - SFB 1608 - 501798263 and KASTEL Security Research Labs, Karlsruhe.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Antoniol, G., Cleland-Huang, J., Hayes, J.H., Vierhauser, M.: Grand Challenges of Traceability: The Next Ten Years. arXiv:1710.03129 [cs] (Oct 2017)
2. Assawamekin, N., Sunetnanta, T., Pluempitiwiriyawej, C.: Ontology-based multiperspective requirements traceability framework. Knowl Inf Syst **25**(3), 493–522 (Dec 2010). https://doi.org/10.1007/s10115-009-0259-2
3. Chen, J., Hu, X., Li, Z., Gao, C., Xia, X., Lo, D.: Code Search is All You Need? Improving Code Suggestions with Code Search. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. pp. 1–13. ACM, Lisbon Portugal (Apr 2024). https://doi.org/10.1145/3597503.3639085
4. Chen, L., Wang, D., Wang, J., Wang, Q.: Enhancing Unsupervised Requirements Traceability with Sequential Semantics. In: 2019 26th Asia-Pacific Software Engineering Conference (APSEC). pp. 23–30 (Dec 2019). https://doi.org/10.1109/APSEC48747.2019.00013
5. Cleland-Huang, J., Gotel, O., Zisman, A., et al.: Software and Systems Traceability, vol. 2. Springer (2012). https://doi.org/10.1007/978-1-4471-2239-5
6. Cleland-Huang, J., Vierhauser, M., Bayley, S.: Dronology: an incubator for cyber-physical systems research. In: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results. pp. 109–112. ICSE-NIER '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183399.3183408
7. CoEST: Center of Excellence for Software & Systems Traceability, http://sarec.nd.edu/coest/
8. Cohen, J.: Statistical Power Analysis for the Behavioral Sciences. Routledge (May 2013). https://doi.org/10.4324/9780203771587

9. Dell'Anna, D., Aydemir, F.B., Dalpiaz, F.: Evaluating classifiers in SE research: The ECSER pipeline and two replication studies. Empirical Software Engineering **28**(1) (Nov 2022). https://doi.org/10.1007/s10664-022-10243-1
10. ECSS: ECSS-E-40C: Principles and requirements applicable to space software engineering (2009)
11. Fuchß, D., Hey, T., Keim, J., Liu, H., Ewald, N., Thirolf, T., Koziolek, A.: LiSSA: Toward Generic Traceability Link Recovery through Retrieval-Augmented Generation. In: Proceedings of the IEEE/ACM 47th International Conference on Software Engineering. ICSE '25 (2025)
12. Gao, H., Kuang, H., Sun, K., Ma, X., Egyed, A., Mäder, P., Rong, G., Shao, D., Zhang, H.: Using Consensual Biterms from Text Structures of Requirements and Code to Improve IR-Based Traceability Recovery. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, ACM (Jan 2023). https://doi.org/10.1145/3551349.3556948
13. Gotel, O., Cleland-Huang, J., Hayes, J.H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J.: The Grand Challenge of Traceability (v1.0). In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) Software and Systems Traceability, pp. 343–409 (2012). https://doi.org/10.1007/978-1-4471-2239-5_16
14. Guo, J., Cheng, J., Cleland-Huang, J.: Semantically Enhanced Software Traceability Using Deep Learning Techniques. In: Proceedings of the 39th International Conference on Software Engineering. pp. 3–14. ICSE '17, IEEE Press, Piscataway, NJ, USA (2017). https://doi.org/10.1109/ICSE.2017.9
15. Hayes, J.H., Dekhtyar, A., Osborne, J.: Improving requirements tracing via information retrieval. In: Proceedings. 11th IEEE International Requirements Engineering Conference, 2003. pp. 138–147 (Sep 2003). https://doi.org/10.1109/ICRE.2003.1232745
16. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. IEEE Trans. Softw. Eng. **32**(1), 4–19 (Jan 2006). https://doi.org/10.1109/TSE.2006.3
17. Hey, T., Chen, F., Weigelt, S., Tichy, W.F.: Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 12–22 (Sep 2021). https://doi.org/10.1109/ICSME52107.2021.00008
18. Hey, T., Fuchß, D., Keim, J., Koziolek, A.: Replication Package for "Requirements Traceability Link Recovery via Retrieval-Augmented Generation" (Jan 2025). https://doi.org/10.5281/zenodo.14779457
19. Hey, T., Keim, J., Corallo, S.: Requirements Classification for Traceability Link Recovery. In: 2024 IEEE 32nd International Requirements Engineering Conference (RE). pp. 155–167 (Jun 2024). https://doi.org/10.1109/RE59067.2024.00024
20. Lan, J., Gong, L., Zhang, J., Zhang, H.: BTLink : Automatic link recovery between issues and commits based on pre-trained BERT model. Empirical Software Engineering **28**(4), 103 (Jul 2023). https://doi.org/10.1007/s10664-023-10342-7
21. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 9459–9474. Curran Associates, Inc. (2020)
22. Liang, H., Hang, D., Li, X.: Modeling function-level interactions for file-level bug localization. Empirical Software Engineering **27**(7) (Oct 2022). https://doi.org/10.1007/s10664-022-10237-z

23. Lin, J., Liu, Y., Zeng, Q., Jiang, M., Cleland-Huang, J.: Traceability Transformed: Generating More Accurate Links with Pre-Trained BERT Models. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 324–335 (May 2021). https://doi.org/10.1109/ICSE43902.2021.00040
24. Lin, J., Poudel, A., Yu, W., Zeng, Q., Jiang, M., Cleland-Huang, J.: Enhancing Automated Software Traceability by Transfer Learning from Open-World Data (arXiv:2207.01084) (Jul 2022)
25. Lohar, S., Amornborvornwong, S., Zisman, A., Cleland-Huang, J.: Improving Trace Accuracy Through Data-driven Configuration and Composition of Tracing Features. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. pp. 378–388. ESEC/FSE 2013, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2491411.2491432
26. Luo, Z., Wang, W., Cen, C.: Improving Bug Localization with Effective Contrastive Learning Representation. IEEE Access pp. 32523–32533 (2022). https://doi.org/10.1109/ACCESS.2022.3228802
27. Mahmoud, A., Niu, N.: On the role of semantics in automated requirements tracing. Requirements Eng **20**(3), 281–300 (Sep 2015). https://doi.org/10.1007/s00766-013-0199-y
28. Mills, C., Escobar-Avila, J., Haiduc, S.: Automatic Traceability Maintenance via Machine Learning Classification. In: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 369–380 (Sep 2018). https://doi.org/10.1109/ICSME.2018.00045
29. Mills, C., Escobar-Avila, J., Bhattacharya, A., Kondyukov, G., Chakraborty, S., Haiduc, S.: Tracing with Less Data: Active Learning for Classification-Based Traceability Link Recovery. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 103–113 (Sep 2019). https://doi.org/10.1109/ICSME.2019.00020
30. Preda, A.R., Mayr-Dorn, C., Mashkoor, A., Egyed, A.: Supporting High-Level to Low-Level Requirements Coverage Reviewing with Large Language Models. In: 21st International Conference on Mining Software Repositories. pp. 242–253. ACM, Lisbon Portugal (Apr 2024). https://doi.org/10.1145/3643991.3644922
31. Rierson, L.: Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance. CRC Press (Jan 2013)
32. Rodriguez, A.D., Cleland-Huang, J., Falessi, D.: Leveraging Intermediate Artifacts to Improve Automated Trace Link Retrieval. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 81–92 (Sep 2021). https://doi.org/10.1109/ICSME52107.2021.00014
33. Rodriguez, A.D., Dearstyne, K.R., Cleland-Huang, J.: Prompts matter: Insights and strategies for prompt engineering in automated software traceability. In: 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW). pp. 455–464 (2023). https://doi.org/10.1109/REW57809.2023.00087
34. RTCA/EUROCAE: DO-178B/ED-12B: Software considerations in airborne systems and equipment certification (2000)
35. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131 (2008). https://doi.org/10.1007/s10664-008-9102-8
36. Sallou, J., Durieux, T., Panichella, A.: Breaking the silence: the threats of using llms in software engineering. In: Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results. p. 102–106. ICSE-NIER'24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3639476.3639764

37. Schlutter, A., Vogelsang, A.: Trace Link Recovery using Semantic Relation Graphs and Spreading Activation. In: 2020 IEEE 28th International Requirements Engineering Conference (RE). pp. 20–31 (Aug 2020). https://doi.org/10.1109/RE48521.2020.00015
38. Schlutter, A., Vogelsang, A.: Improving Trace Link Recovery Using Semantic Relation Graphs and Spreading Activation. In: Dalpiaz, F., Spoletini, P. (eds.) Requirements Engineering: Foundation for Software Quality. pp. 37–53. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-73128-1_3
39. Shin, Y., Hayes, J.H., Cleland-Huang, J.: Guidelines for Benchmarking Automated Software Traceability Techniques. In: 2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability. pp. 61–67 (May 2015). https://doi.org/10.1109/SST.2015.13
40. Tian, F., Wang, T., Liang, P., Wang, C., Khan, A.A., Babar, M.A.: The impact of traceability on software maintenance and evolution: A mapping study. Journal of Software: Evolution and Process **33**(10) (2021). https://doi.org/10.1002/smr.2374
41. Wang, W., Niu, N., Liu, H., Niu, Z.: Enhancing Automated Requirements Traceability by Resolving Polysemy. In: 2018 IEEE 26th International Requirements Engineering Conference (RE) (Aug 2018). https://doi.org/10.1109/RE.2018.00-53
42. Zhao, T., Cao, Q., Sun, Q.: An Improved Approach to Traceability Recovery Based on Word Embeddings. In: 2017 24th Asia-Pacific Software Engineering Conference (APSEC). pp. 81–89 (Dec 2017). https://doi.org/10.1109/APSEC.2017.14
43. Zou, X., Settimi, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: A study of term-based enhancement methods. Empir Software Eng **15**(2), 119–146 (Apr 2010). https://doi.org/10.1007/s10664-009-9114-z